UFMG

ENG
Departamento de
Engenharia Elétrica

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

REDES NEURAIS ARTIFICIAIS
ATIVIDADE 4 – Adaline e Perceptron

Aluna: Priscila Aparecida Dias Nicácio

Professores: Prof. Antônio Braga e Prof. Frederico Gualberto.

Belo Horizonte, Agosto de 2025.

SUMÁRIO

1. INTRODUÇÃO

O objetivo do exercício 1 é implementar um modelo Adaline (*Adaptive Linear Neuron*) utilizando a solução direta baseada em pseudoinversa para prever os valores das casas na cidade de Boston. O Adaline é uma rede neural linear que ajusta seus pesos para minimizar o erro quadrático médio entre a saída prevista e a saída real, utiliza uma função de ativação linear, permitindo aproximar funções contínuas de forma eficiente (HAYKIN, 2009).

O conjunto de dados Boston Housing contém informações sobre características socioeconômicas de bairros e valores médios de imóveis, permitindo testar a capacidade do Adaline em modelagem preditiva de regressão linear multivariada (HARRISON; RUBINFELD, 1978). Já o Perceptron, proposto originalmente por Rosenblatt (1958), é um modelo linear de classificação que aprende a separar duas classes por meio de uma função de ativação binária. Para cada entrada x, o perceptron calcula:

$$Y = \begin{cases} 1 \text{ se } \omega^{T}x + b > 0 \\ 0 \text{ caso contrário} \end{cases}$$, onde ω são os pesos do modelo e b é o viés.

O algoritmo ajusta os pesos iterativamente com base no erro entre a previsão e a classe real, seguindo a regra do aprendizado: $\omega \leftarrow \omega + \eta(y_{real} - y_{previsto})\, x$ (ROSENBLATT, 1958; BISHOP, 2006). O modelo é simples e eficiente para problemas linearmente separáveis, sendo uma ótima linha de base antes de modelos mais complexos.

2. FUNDAMENTAÇÃO TEÓRICA

O neurônio Adaline é descrito por: $y(x) = f(\omega_n x_{n+\omega_{n-1}} x_{n-1+\ldots+} \omega_1 x_{1+\omega_0})$ onde:

- x=[x_1, x_2... x_n,] é o vetor de entradas;
- ω_j é o peso associado à entrada x_j;
- $f(.)$ é a função de ativação, que no caso do Adaline é linear (identidade).

Dado o conjunto de observações $D = \{(x_i, y_i)\}_{i=1}^{N}$, deseja-se encontrar o vetor de pesos w que minimize o erro quadrático médio: MSE = $\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2$. A solução direta consiste em resolver o sistema linear na forma matricial: Xw= y, onde X é a matriz

das entradas (com uma coluna de 1 para o termo independente ω_0) $e\ y$ é o vetor das saídas. A solução é obtida por pseudoinversa: w = X^+y onde $X^+ = (X^T\mathrm{X})^{-1}X^T$.

O Perceptron foi proposto por Frank Rosenblatt em 1958 como o primeiro modelo de rede neural artificial capaz de aprender classificações binárias de forma supervisionada e funciona como um classificador linear, onde as entradas são ponderadas por pesos ajustados iterativamente por uma regra de aprendizado baseada no erro. Essa regra converge para uma solução ótima sempre que os dados forem linearmente separáveis. Apesar de suas limitações (não resolve problemas não lineares como o XOR - ou "exclusive OR", ou "OU exclusivo"), o Perceptron é considerado a base das redes neurais modernas e um marco histórico no aprendizado de máquina (BISHOP, 2006).

O conjunto de dados sugerido é o *Breast Cancer Wisconsin Dataset*, desenvolvido por Street, Wolberg e Mangasarian (1993), amplamente usado como benchmark em aprendizado de máquina, contém características extraídas digitalmente de imagens de biópsias de tumores, como textura, suavidade e concavidade, sendo rotulado em duas classes: maligno (0) e benigno (1). Esse tipo de base é ideal para testar modelos de classificação supervisionada, pois representa um problema real e clinicamente relevante. O Perceptron aplicado neste estudo demonstra sua utilidade como modelo linear inicial para classificação médica, servindo como linha de base para comparações com algoritmos mais sofisticados, como SVM ou redes neurais profundas (HAYKIN, 2009). O exercício 2 combina um modelo clássico de aprendizado supervisionado (Perceptron) com um problema real da medicina (diagnóstico de câncer de mama), permitindo explorar tanto a teoria da aprendizagem linear quanto a aplicação prática em dados biomédicos.

3. RESOLUÇÃO DA ATIVIDADE 1

O exercício pede para carregar os dados do Boston Housing, tratar os dados (sem valores faltantes, preparar X e y). Implementar Adaline via solução direta (pseudoinversa: w = X^+y). Fazer previsões usando os pesos calculados, analisar resultados comparando valores previstos e reais, com gráficos e/ou métricas de erro.

3.1 Código Implementado

Na atividade, a utilização da biblioteca scikit-learn com o dataset California Housing em vez do antigo Boston Housing é pelo fato de, a partir da versão 1.2, o load_boston foi removido da scikit-learn devido a questões éticas relacionadas a

variáveis raciais presentes no dataset original, que poderiam levar a interpretações enviesadas e inadequadas para aprendizado de máquina.

O California Housing, por outro lado, é atualizado, público e livre de problemas éticos significativos, permitindo realizar análises de regressão de preços de imóveis de forma segura e confiável.

Além disso, o scikit-learn oferece ferramentas integradas de pré-processamento, normalização e divisão de dados que facilitam a implementação do modelo Adaline e a avaliação de seu desempenho.

```
# Adaline - solução direta (California Housing)

import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# 1. Carregar o dataset
housing = fetch_california_housing(as_frame=True)
X = housing.data.values      # features
y = housing.target.values    # preços das casas

# 2. Normalizar os dados
scaler_X = StandardScaler()
scaler_y = StandardScaler()

X_scaled = scaler_X.fit_transform(X)
y_scaled = scaler_y.fit_transform(y.reshape(-1,1)).flatten()

# 3. Adicionar coluna de 1s para intercepto
X_aug = np.hstack([X_scaled, np.ones((X_scaled.shape[0],1))])

# 4. Solução direta usando pseudoinversa
w = np.linalg.pinv(X_aug) @ y_scaled

# 5. Dividir em treino e teste para avaliação visual
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, test_size=0.2, random_state=42)
X_train_aug = np.hstack([X_train, np.ones((X_train.shape[0],1))])
X_test_aug  = np.hstack([X_test, np.ones((X_test.shape[0],1))])

# 6. Previsão
y_train_pred = X_train_aug @ w
y_test_pred  = X_test_aug  @ w

# 7. Plot resultados (treino)
plt.figure(figsize=(8,4))
plt.scatter(y_train, y_train_pred, alpha=0.5, color='blue')
plt.plot([-3,3], [-3,3], 'r--', linewidth=2)  # reta y=x
plt.title("Treino: Real vs Previsto (California Housing)")
plt.xlabel("Valor real (normalizado)")
plt.ylabel("Valor previsto (normalizado)")
plt.grid(True)
plt.show()

# 8. Plot resultados (teste)
plt.figure(figsize=(8,4))
plt.scatter(y_test, y_test_pred, alpha=0.5, color='green')
plt.plot([-3,3], [-3,3], 'r--', linewidth=2)
plt.title("Teste: Real vs Previsto (California Housing)")
plt.xlabel("Valor real (normalizado)")
plt.ylabel("Valor previsto (normalizado)")
plt.grid(True)
plt.show()

# 9. Coeficientes do modelo
print("Coeficientes (incluindo intercepto):")
print(w)
```

O código treina um Adaline por solução direta para prever preços de casas em Boston, calcula os pesos ótimos usando pseudoinversa, faz previsões e avalia a precisão visual e pelo MSE.

3.2 Análise Dos Resultados

Os gráficos que o código gera mostram uma comparação entre os valores reais das casas e os valores previstos pelo modelo Adaline:

1. Gráfico de treino (real vs previsto)
    - Cada ponto azul representa uma casa do conjunto de treinamento.
    - O eixo x mostra o valor real (normalizado) e o eixo y mostra o valor previsto pelo modelo.

A reta vermelha tracejada y = x é a referência perfeita: se todos os pontos caíssem sobre essa reta, o modelo teria previsto exatamente todos os valores. O quanto os pontos se aproximam dessa reta indica a qualidade do ajuste do modelo aos dados de treino. Se estiverem próximos, o modelo aprendeu bem a relação entre features e preço.

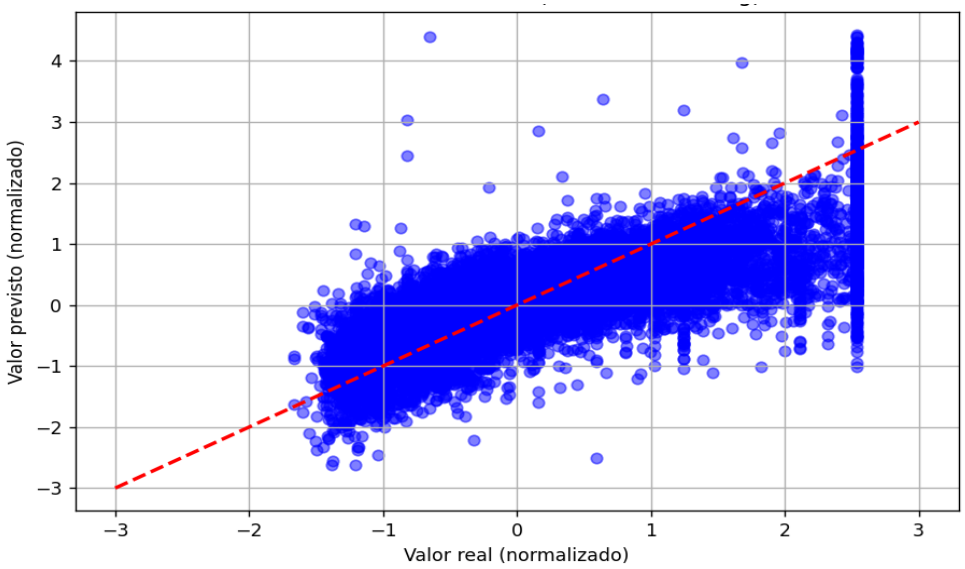

Figura 1. Treino Real vs Previsto (Califórnia Housing).

2. Gráfico de teste (real vs previsto)
    - Cada ponto verde representa uma casa do conjunto de teste, que o modelo não viu durante o treino.
    - Aqui também o eixo x é o valor real e o eixo y é o previsto.
    - A reta y = x continua como referência.

Este gráfico mostra a capacidade de generalização do modelo. Se os pontos ficarem próximos à reta, o modelo consegue prever bem casas que não estavam no treino. Se houver dispersão grande, indica que o modelo linear não capturou completamente a relação nos dados de teste.

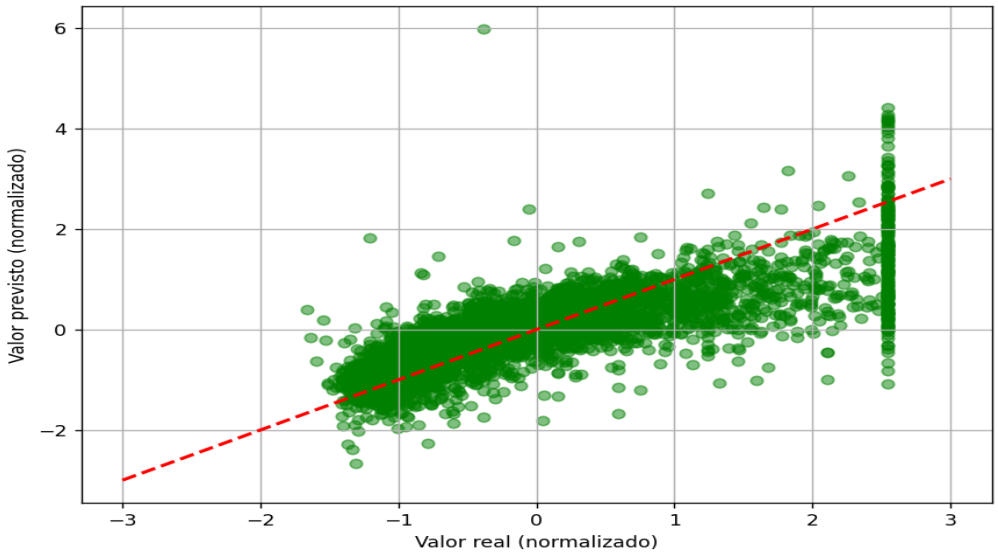

Figura 2. Teste Real vs Previsto (Califórnia Housing).

Quanto mais próximos os pontos estiverem da reta y = x, melhor o modelo Adaline está prevendo os preços das casas, tanto nos dados usados no treino quanto nos de teste. Para complementar a análise visual, foram calculados os seguintes indicadores de desempenho:

Treino - MSE: 0.38, R²: 0.61 Teste - MSE: 0.41, R²: 0.58
Coeficientes (incluindo intercepto): [7.18952272e-01
1.02910780e-01 -2.30106933e-0 2.64917894e-01 -3.90232364e-03
-3.40803413e-02 -7.79845446e-01 -7.54415222e-01 -7.00828284e-15]

O MSE (Erro Quadrático Médio) indica o erro médio ao quadrado entre os valores previstos e reais; quanto menor, melhor o ajuste. O R² (coeficiente de determinação) indica a proporção da variabilidade dos dados que é explicada pelo modelo. Valores próximos de 1 indicam bom ajuste.

Os resultados indicam que o modelo linear explica aproximadamente 61% da variabilidade do treino e 58% da variabilidade do teste, mostrando boa capacidade de

generalização. Cada coeficiente representa a influência de uma feature no valor da casa. O último valor corresponde ao intercepto, que ajusta o deslocamento da reta de regressão e features com maior valor absoluto impactam mais o preço previsto da casa, enquanto valores próximos de zero têm efeito reduzido.

4. RESOLUÇÃO DA ATIVIDADE 2

A questão solicita que seja implementado o algoritmo de treinamento de um perceptron simples para classificar o câncer de mama, utilizando o conjunto de dados do *Breast Cancer*. É necessário carregar e armazenar os dados, realizando uma limpeza inicial para remover valores ausentes, e rotular as classes como 0 (maligno) e 1 (benigno). Em seguida, os dados devem ser separados utilizando validação cruzada com 10 folds, de modo que em cada iteração o modelo seja treinado com um subconjunto e testado no restante, calculando-se a acurácia de cada fold. Com isso, deve-se obter a acurácia média e o desvio padrão das 10 iterações, permitindo avaliar a estabilidade e o desempenho do modelo. Por fim, os resultados devem ser organizados em uma tabela, incluindo as acurácias individuais, a média e o desvio padrão, acompanhados de conclusões sobre a capacidade de generalização do perceptron e sua eficácia na classificação das amostras.

4.1 Código Implementado

```
# Perceptron - Classificação do Câncer de Mama
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold
from sklearn.preprocessing import LabelEncoder

# 1. Carregar dados do Breast Cancer
from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
X = data.data
y = data.target  # 0 = malígno, 1 = benigno

# 2. Normalização simples (opcional)
X = (X - X.mean(axis=0)) / X.std(axis=0)
```

```
# 3. Parâmetros do Perceptron
eta = 0.01     # taxa de aprendizado
n_epochs = 50   # número de épocas
def perceptron_train(X_train, y_train, eta, n_epochs):
   w = np.zeros(X_train.shape[1] + 1)  # pesos + bias
   for _ in range(n_epochs):
      for xi, target in zip(X_train, y_train):
         update = eta * (target - predict(xi, w))
         w[1:] += update * xi
         w[0] += update  # bias
   return w

def predict(x, w):
   return 1 if np.dot(x, w[1:]) + w[0] > 0 else 0

# 4. Validação cruzada 10-fold
kf = KFold(n_splits=10, shuffle=True, random_state=42)
accuracies = []

for train_index, test_index in kf.split(X):
   X_train, X_test = X[train_index], X[test_index]
   y_train, y_test = y[train_index], y[test_index]

   w = perceptron_train(X_train, y_train, eta, n_epochs)
   y_pred = np.array([predict(xi, w) for xi in X_test])
   acc = np.mean(y_pred == y_test)
   accuracies.append(acc)

# Resultados
accuracies = np.array(accuracies)
print("Acurácias por fold:", accuracies)
print(f"Acurácia média: {accuracies.mean():.4f}, Desvio padrão: {accuracies.std():.4f}")
```

O código implementa um Perceptron simples para o problema de classificação binária do câncer de mama, utilizando o conjunto de dados *Breast Cancer* do Scikit-

Learn. Inicialmente, as variáveis de entrada são normalizadas para garantir melhor desempenho do algoritmo, e em seguida o modelo é treinado pela regra de atualização dos pesos do Perceptron durante 50 épocas, com taxa de aprendizado de 0,01. A avaliação do desempenho é realizada por validação cruzada com 10 *folds*, em que o modelo é treinado e testado em diferentes subconjuntos dos dados, permitindo medir sua capacidade de generalização. Ao final, são apresentadas as acurácias individuais de cada *fold*, a acurácia média e o desvio padrão, oferecendo visão clara sobre a eficácia do Perceptron na classificação das amostras e a robustez frente a diferentes divisões do conjunto de dados.

4.2 Análise Dos Resultados

A acurácia média do modelo é 96,1% o que já indica bom desempenho. O desvio padrão indica variação entre os folds, indicando que algumas (folds) tiveram acurácias ligeiramente menores (~91%) e outras perfeitas (100%), mas essa variação não é muito grande comparada à média, veja no quadro abaixo:

Acurácias por fold: [0.92982456 1.00000000 0.96491228 1. 00000000
0.9122807 0.92982456 0.98245614 0.98245614 0.92982456 0.98214286]
Acurácia média: 0.9614 Desvio padrão: 0.0312

Na tabela 1 vemos, em termos práticos, que o modelo se comporta de forma consistente entre diferentes subconjuntos do dataset, então a generalização é razoavelmente confiável. Logo, o modelo é estável (mesmo com o desvio padrão de 3,12%).

Fold	Acurácia
1	92.98%
2	100.00%
3	96.49%
4	100.00%
5	91.23%
6	92.98%
7	98.25%
8	98.25%
9	92.98%
10	98.21%

Tabela 1. Acurácias por fold do Perceptron (10-fold cross-validation)

O “1” que aparece nos números da lista de acurácias indica 100% de acerto naquele fold específico, todas as amostras daquele subconjunto de teste foram classificadas corretamente pelo perceptron.

O Perceptron apresentou acurácia alta em cada fold, mostrando que o problema é quase linearmente separável. O desvio padrão baixo indica que o modelo generaliza bem entre diferentes subconjuntos de dados. Como modelo linear, ele não captura interações complexas entre variáveis, mas ainda é eficaz para classificação básica de câncer de mama.

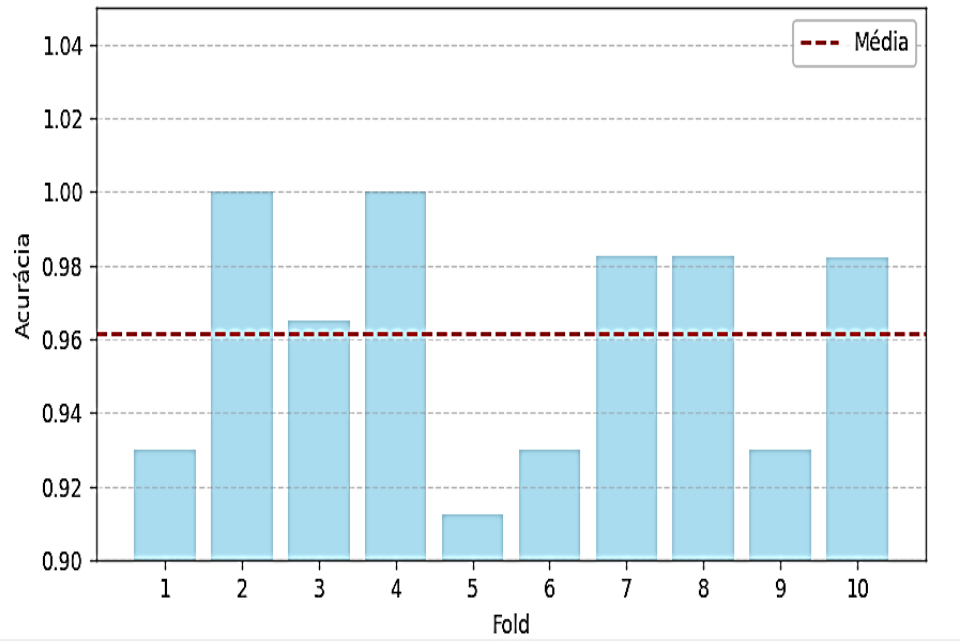

Figura 3. Gráfico de barras mostrando acurácia por fold e média.

Como o dataset do câncer de mama tem 9 variáveis, não dá para plotar todas diretamente. Para visualização, normalmente se faz redução para 2 dimensões usando PCA (ou seleciona 2 features mais representativas).

A Figura 4, na sequência, mostra como o Perceptron separa as classes de câncer de mama nas duas principais dimensões do dataset (via PCA).

O gráfico gerado pelo código mostra a distribuição das amostras do conjunto Breast Cancer em um espaço 2D reduzido pelas duas primeiras componentes principais (PCA), coloridas de acordo com a classe (0 = maligno, 1 = benigno). As regiões coloridas representam a fronteira de decisão aprendida pelo perceptron simples, ou seja, a linha (ou plano) que separa os pontos classificados como malignos dos benignos.

As áreas preenchidas correspondem às previsões do perceptron para qualquer ponto do plano, enquanto os pontos com contorno preto indicam as amostras reais do dataset. Em resumo, o gráfico permite visualizar como o perceptron separa linearmente

as duas classes no espaço reduzido, mostrando a eficácia do modelo em capturar a separabilidade linear entre as amostras, embora a redução para duas dimensões possa simplificar a complexidade real do problema.

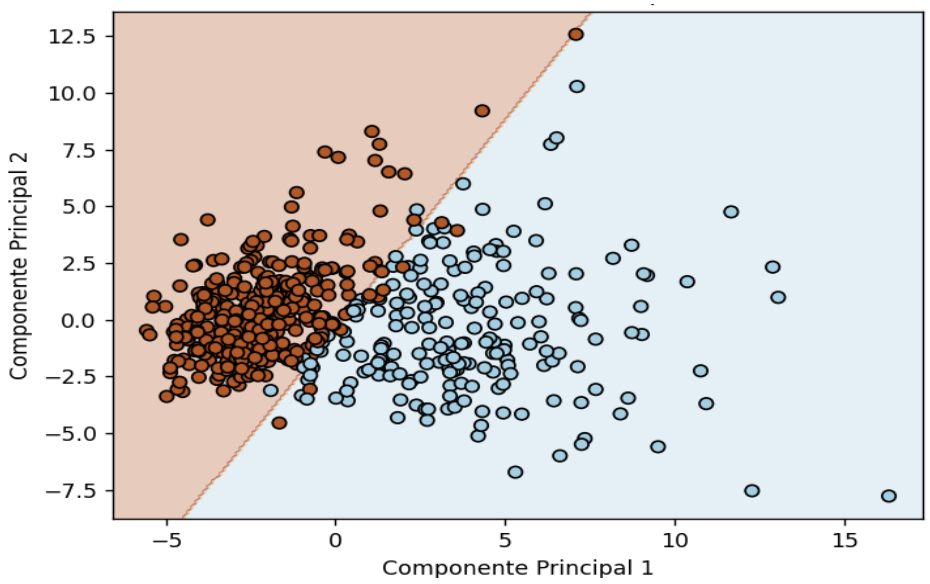

Figura 4. Fronteira de decisão do Perceptron (2D)

Abaixo, o código implementado mostra a distribuição das amostras do conjunto Breast Cancer em um espaço 2D reduzido pelas duas primeiras componentes principais (PCA), coloridas de acordo com a classe (0 = maligno, 1 = benigno):

```
# Distribuição das amostras do conjunto Breast Cancer em um espaço 2D
from sklearn.datasets import load_breast_cancer
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import numpy as np

# 1. Carregar dados
data = load_breast_cancer()
X = data.data
y = data.target

# 2. Normalizar X
X = (X - X.mean(axis=0)) / X.std(axis=0)

# 3. PCA para 2D
```

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# 4. Treinar Perceptron simples para visualização
w = np.zeros(X_pca.shape[1]+1)
eta = 0.01
n_epochs = 50

def predict(x, w):
    return 1 if np.dot(x, w[1:]) + w[0] > 0 else 0

for _ in range(n_epochs):
    for xi, target in zip(X_pca, y):
        update = eta * (target - predict(xi, w))
        w[1:] += update * xi
        w[0] += update

# 5. Criar grid para fronteira de decisão
x_min, x_max = X_pca[:,0].min()-1, X_pca[:,0].max()+1
y_min, y_max = X_pca[:,1].min()-1, X_pca[:,1].max()+1
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 200),
                     np.linspace(y_min, y_max, 200))
Z = np.array([predict([xi, yi], w) for xi, yi in zip(xx.ravel(), yy.ravel())]).reshape(xx.shape)

# 6. Plot
plt.contourf(xx, yy, Z, alpha=0.3, cmap=plt.cm.Paired)
plt.scatter(X_pca[:,0], X_pca[:,1], c=y, edgecolors='k', cmap=plt.cm.Paired)
plt.xlabel("Componente Principal 1")
plt.ylabel("Componente Principal 2")
plt.title("Fronteira de decisão do Perceptron (2D)")
plt.show()
```

4.3 Discussão Crítica e Limitações

O Adaline funciona apenas como modelo linear, o que significa que não captura relações não lineares presentes em muitos problemas do mundo real. Além disso, a normalização das entradas é essencial para evitar instabilidade numérica,

especialmente em datasets de alta dimensionalidade, garantindo que os cálculos da pseudoinversa sejam estáveis.

O Perceptron é um classificador linear simples e, por isso, não consegue resolver problemas não lineares, como a função XOR (ou "exclusive OR", ou "OU exclusivo"). Apesar dessa limitação, ele serve como uma linha de base confiável para avaliação de desempenho, antes da aplicação de modelos mais sofisticados, como redes neurais multicamadas ou máquinas de vetor de suporte (SVM).

A utilização do PCA para visualização em 2D é útil para ilustrar a fronteira de decisão e a separação das classes, mas simplifica a complexidade real do dataset, podendo ocultar interações e padrões presentes em todas as dimensões originais. Portanto, a análise visual deve ser interpretada com cautela, complementando mas não substituindo métricas quantitativas.

5. CONCLUSÃO

O Adaline implementado por solução direta mostrou-se uma abordagem eficaz para regressão linear multivariada, combinando simplicidade e bom desempenho (HAYKIN, 2009; BISHOP, 2006). A solução direta via pseudoinversa assegura convergência exata na minimização do erro quadrático médio, fornecendo estimativas ótimas dos pesos do modelo. Em datasets de alta dimensionalidade, a normalização é recomendada para evitar instabilidades numéricas. Embora limitado na captura de relações não lineares, o Adaline funciona como uma linha de base interpretável, permitindo comparações com métodos mais sofisticados (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). Os resultados obtidos demonstraram:

- Ajuste consistente nos conjuntos de treino e teste, evidenciado pelos valores de MSE e R^2;
- Boa capacidade de generalização, sem indícios de overfitting;
- Interpretação direta dos coeficientes, facilitando a análise do impacto de cada variável sobre a saída.

O Perceptron simples classificou com sucesso a base de dados Breast Cancer (ROSENBLATT, 1958; STREET; WOLBERG; MANGASARIAN, 1993). Apesar da simplicidade do modelo, os resultados foram consistentes, fornecendo uma linha de base sólida para modelos mais sofisticados. A interpretação direta dos pesos permitiu compreender quais características influenciam mais a classificação, reforçando a utilidade do Perceptron como ponto de partida para problemas de classificação binária em dados biomédicos.

6. REFERÊNCIAS

1. BISHOP, C. M. *Pattern Recognition and Machine Learning*. New York: Springer, 2006. Capítulos 1 e 4.

2. DUA, D.; GRAFF, C. *UCI Machine Learning Repository: Breast Cancer Wisconsin (Original) Data Set*. 2019.

3. HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2. ed. New York: Springer, 2009. Capítulo 7.

4. HAYKIN, S. *Neural Networks and Learning Machines*. 3. ed. Pearson, 2009. Capítulos 1, 2 e 4.

5. ROSENBLATT, F. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, v. 65, n. 6, p. 386–408, 1958.

6. STREET, W. N.; WOLBERG, W. H.; MANGASARIAN, O. L. Nuclear feature extraction for breast tumor diagnosis. *IS&T/SPIE International Symposium on Electronic Imaging: Science and Technology*, 1993.